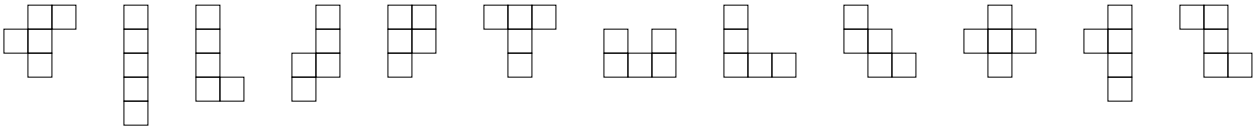
UPMC/Licence/Info/2I013
Pentodroid
Modèle du jeu

Janvier 2017

Un *pentomino* est une figure géométrique constitué de 5 carrés d'égales surfaces assemblés par leurs cotés. Un jeu de pentominos consiste à paver une surface avec ces pièces, sans utiliser 2 fois une même pièce.

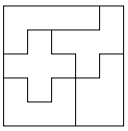
Il y a 12 pentominos de base possibles. On nomme chacun d'eux par une lettre évoquant plus ou moins leur forme.



Les 12 pentominos de base

On peut faire tourner un pentomino de 90° (rotation) ou le retourner (miroir). Ces variations donnent lieu à 63 figures différentes (voir figure 1).

Si l'on considère les pentominos L (variante L1b), X, U (variante Ua), F (variante F2d) et P (variante P1c), on peut les assembler en un carré de 5 sur 5.



Une partie de pentominos se jouera à partir d'un ensemble de (variantes de) pentominos et la taille du rectangle à construire (exprimée en nombre de carrés) avec ces pièces. La partie donnée en exemple est décrite par la donnée de : 5 5 F2d L1b P1c Ua et X.

Le modèle du jeu est une représentation en JAVA d'une partie de pentominos. Il doit permettre de placer un pentamino choisi dans la liste donnée en début de partie sur le rectangle à construire ; de retirer du rectangle une pièce placée, etc. Il est implémenté par une classe JAVA répondant à l'interface `IModel`.

1 Utilitaires

`Pair<T1,T2>` est une classe paramétrée destinée à représenter des couples de valeurs ou d'objets. Le premier est instance de `T1` et le second, de `T2`. Elle fournit :

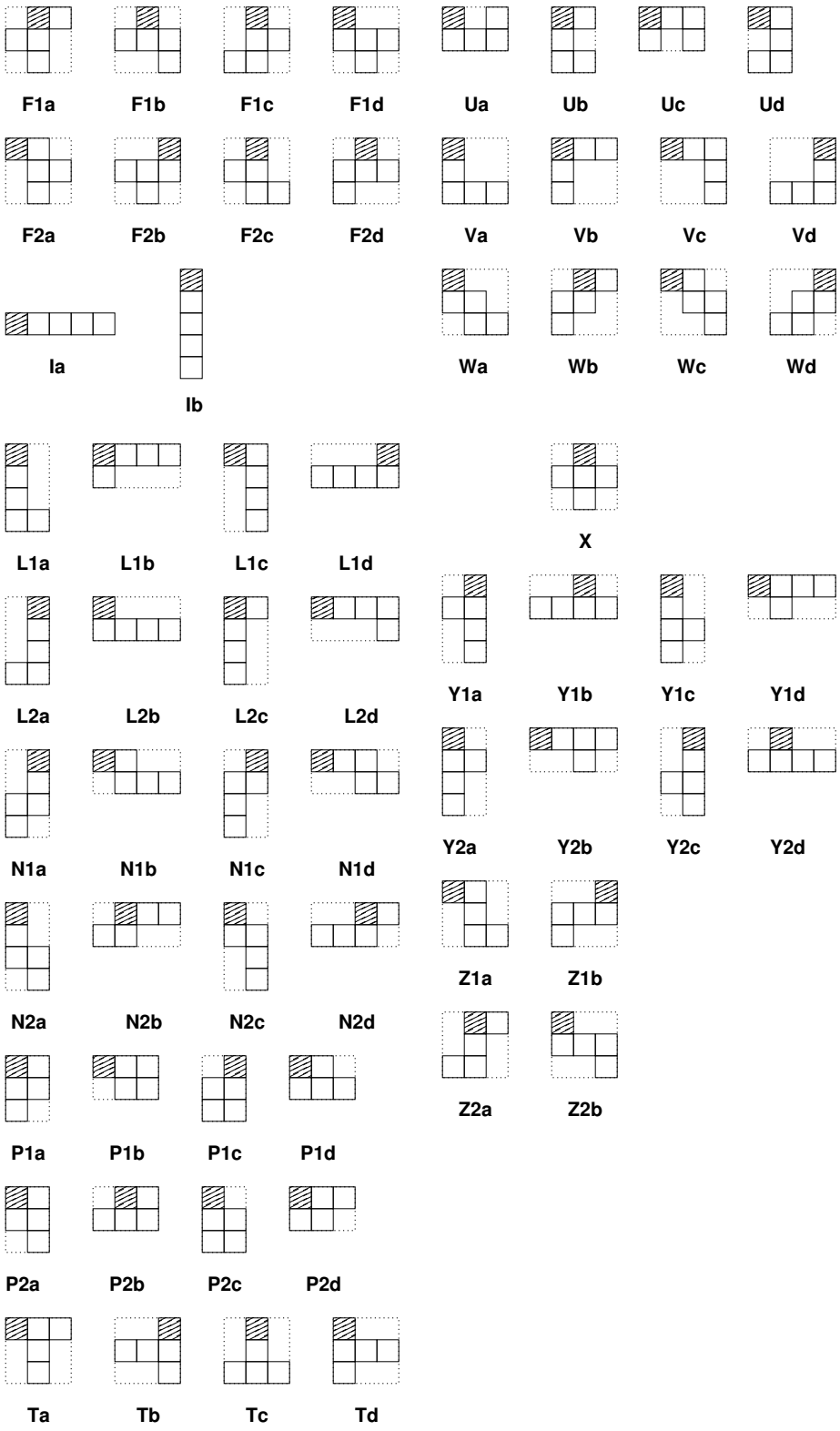


FIGURE 1 – Nomenclature des 63 variantes des pentominos

— le constructeur `Pair(T1 v1, T2 v2)` ;
— la méthode `T1 fst()` pour la première projection ;
— la méthode `T2 snd()` pour la seconde projection.
Cette classe vous est fournie.

`Coordinate` est la classe qui représente les coordonnées (paires d'entiers). Elle hérite de `Pair` et fournit :
— le constructeur `Coordinate(int lig, int col)`. Le premier argument est le numéro de ligne, le second, le numéro de colonne ;
— la méthode `int getLig()` qui donne le numéro de ligne ;
— la méthode `int getCol()` qui donne le numéro de colonne.
Cette classe vous est fournie.

`NotFound` sous classe de `Exception`.
Cette classe vous est fournie.

2 Représentation des pentominos

On représente un pentomino par son nom et une liste de couples d'entiers décrivant la manière dont ses carrés sont agencés. Ces couple d'entiers correspondent aux *coordonnées* du carré dans la grille minimale enveloppant le pentomino. Par exemple, au pentomino `F1a` est associé la liste [(0,1) ; (0,2) ; (1,0) ; (1,1) ; (2,1)]. Les coordonnées sont données dans l'ordre : numéro de ligne, puis numéro de colonne.

Sur la figure 1 l'un des carrés de chaque pièce est hachuré. C'est le carré correspondant à la première coordonnée de la liste qui décrit le pentomino. Nous dirons que ce carré est le *premier carré* du pentomino. Il nous servira de repère pour le placement des pièces.

2.1 Un type énuméré pour les pentominos

L'ensemble des 63 variantes de pentominos est représenté par les valeurs du *type énuméré* `Pentomino`. Le nom des constantes déclarées dans ce type est celui donné dans la figure 1. Les valeurs de ce type répondent à la méthode `ArrayList<Coordinate> shapeOf()` qui donne la liste des 5 coordonnées décrivant l'assemblage des 5 carrés du pentomino.
Ce type vous est fourni.

En JAVA on peut obtenir une valeur appartenant à un type énuméré à partir d'une chaîne de caractères. Par exemple, l'expression `Enum.valueOf("F1a")` a pour valeur `Pentomino.F1a`.

3 Le modèle

Le modèle d'une partie de pentomino est donc réalisé par la classe `Model` qui implémente *l'interface* `IModel`. Cette interface déclare :
— la méthode `int width()` qui donne la largeur (nombre de colonnes) du rectangle à construire ;
— la méthode `int height()` qui donne la hauteur (nombre de lignes) du rectangle à construire ;
— la méthode `boolean valid(Coordinate pos)` qui vaut `true` si la position `pos` est dans les limites du rectangle. On dit alors que la position est valide. L'invocation de la méthode donne `false` sinon ;
— la méthode `boolean free(Coordinate pos)` qui vaut `true` si la position `pos` est une position valide non couverte du rectangle à construire, et `false` sinon. Lorsque la méthode donne `true`, on dit que la position est libre ;

— la méthode `boolean achieved()` qui vaut `true` si la construction du rectangle est achevée (fin de partie) et `false`, sinon ;
— la méthode `isPlaced(Pentomino id)` qui vaut `true` si le pentomino `id` est placé sur le rectangle à construire ;
— la méthode `Pentomino get(Coordinate pos) throws NotFound` qui donne l'identifiant du pentomino couvrant la position `pos` du rectangle à construire ou déclanche l'exception `NotFound` lorsque la position est libre ou non valide ;
— la méthode `boolean canPut(Pentomino id, Coordinate pos)` qui vaut `true` si le pentomino `id` peut être placé à la position `pos` du retangle à construire et `false`, sinon. La position `pos` désigne sur le rectangle à construire la position qu'occupera le premier carré (au sens de 2) du pentomino `id`. Un pentomino peut être placé à une position si et seulement si

1. Le pentomino n'est pas déjà placé.
2. La position est valide.
3. Toutes les positions du rectangle à construire que couvrira le pentomino sont libres.

— la méthode `void put(Pentomino id, Coordinate pos)` ajoute au rectangle en construction le pentomino `id`. La position `pos` désigne sur le rectangle à construire la position qu'occupera le premier carré (au sens de 2) du pentomino `id` ;
— la méthode `void remove(Pentomino id)` retire le pentomino `id` du rectangle à construire ;
— la méthode `ArrayList<Pair<Pentomino,ArrayList<Coordinate>>> placedPentominos()` qui donne la liste des pentominos placés sur le rectangle à construire associés à la liste des positions qu'ils couvrent.

La classe `Model` doit de surcroît fournir le constructeur : `Model(int width, int height)` Où `width` et `height` donnent la largeur et la hauteur du rectangle à construire. À la création, toutes les cases du rectangle à construire doivent être libres, donc, aucune pièce ne doit y être placée. Les coordonnées des cases vont de $(0, 0)$ à $(h - 1, w - 1)$ où h est le nombre de lignes (hauteur) et w le nombre de colonnes (largeur).